

# OpenCV Tutorial

Hemanth Haridas

# Tasks

The first step after establishing a working environment is to begin manipulating images. This tutorial will give an introduction to the usage of some basic functions.

## Steps Performed

Load an Image

Calculate Histogram Values

Calculate Basic Statistics Using Histogram Information

For explanations on any functions used here see the OpenCV documentat.

# Loading the Image

OpenCV makes it relatively easy to load images. There are several syntax variations that simply take in the path/file name. One is presented here.

## Load the File

Specify a File to be Loaded

Use the *cvLoadImage* function to assign the image to an *IplImage* pointer

```
//the name of the image being loaded
char* imageName = "Critters_00005.JPG";
//Load the image and make sure that it loads correctly
IplImage* im = cvLoadImage(imageName, -1);
if( im == 0 ) {
    //Drop out if the image isn't found
    std::cerr << "Failed to load: " << imageName << std::endl;
    return 1;
}
```

OpenCV uses an *IplImage* to represent image internally.

# Specifying a Working Region

In order to work with a histogram the image will have to be converted to a single plane.

## Create the Grayscale Image

Create an Image of a Single Plane

Convert the Image to Gray

Specify a Rectangular Region of Interest (ROI) and apply it to the image

```
//Create a single planed image of the same size as the original
IplImage* grayImage = cvCreateImage(cvSize(im->width,im->height),
                                     IPL_DEPTH_8U, 1);

//convert the original image to gray
cvCvtColor(im, grayImage, CV_BGR2GRAY);
//create a rectangular area to evaluate
CvRect rect = cvRect(0, 0, 500, 600 );
//apply the rectangle to the image and establish a region of interest
cvSetImageROI(grayImage, rect);
```

The *cvCvtColor* function can be used to convert images to one of several color spaces.

To restore the region of interest to the whole image the function *cvResetImageROI* is used

# Perform Initial Histogram Calculations

OpenCV provides built-in functions to work with histograms.

## Create the Histogram Data

Create a Histogram Image and a Histogram

Calculate the Histogram

Grab Min/Max Values

Set Up Factors For Visualization

```
//create an image to hold the histogram
IplImage* histImage = cvCreateImage(cvSize(320,200), 8, 1);
//create a histogram to store the information from the image
CvHistogram* hist =
    cvCreateHist(1, &hist_size, CV_HIST_ARRAY, ranges, 1);
//calculate the histogram and apply to hist
cvCalcHist( &grayImage, hist, 0, NULL );

//grab the min and max values and their indices
cvGetMinMaxHistValue( hist, &min_value, &max_value, &min_idx, &max_idx);
//scale the bin values so that they will fit in the image representation
cvScale( hist->bins, hist->bins, ((double)histImage->height)/max_value, 0 );

//set all histogram values to 255
cvSet( histImage, cvScalarAll(255), 0 );
//create a factor for scaling along the width
bin_w = cvRound((double)histImage->width/hist_size);
```

# Prepare Visualization/Perform Calculations

Here we will iterate across the histogram bins and apply the values to the image while calculating the statistics.

## Draw Values on Image

Use *cvRectangle* to draw.

Get Values/Perform Calculations

```
for( i = 0; i < hist_size; i++ ) {
    //draw the histogram data onto the histogram image
    cvRectangle( histImage, cvPoint(i*bin_w, histImage->height),
                cvPoint((i+1)*bin_w,
                        histImage->height - cvRound(cvGetReal1D(hist->bins,i))),
                cvScalarAll(0), -1, 8, 0 );
    //get the value at the current histogram bucket
    float* bins = cvGetHistValue_1D(hist,i);
    //increment the mean value
    mean += bins[0];
}
//finish mean calculation
mean /= hist_size;
//go back through now that mean has been calculated in order to calculate variance
for( i = 0; i < hist_size; i++ ) {
    float* bins = cvGetHistValue_1D(hist,i);
    variance += pow((bins[0] - mean),2);
}
//finish variance calculation
variance /= hist_size;
```

# Display Results

This segment displays the visual and textural results.

## Display

Output Statistics

## Show Images

*cvNamedWindow* creates a container. The first parameter is the name and the second declares if the container is to expand to fit the contents.

Hold For Input. Passing the parameter "0" waits for a keypress.

```
std::cout << "Histogram Mean: " << mean << std::endl;
std::cout << "Variance: " << variance << std::endl;
std::cout << "Standard Deviation: " << sqrt(variance) << std::endl;

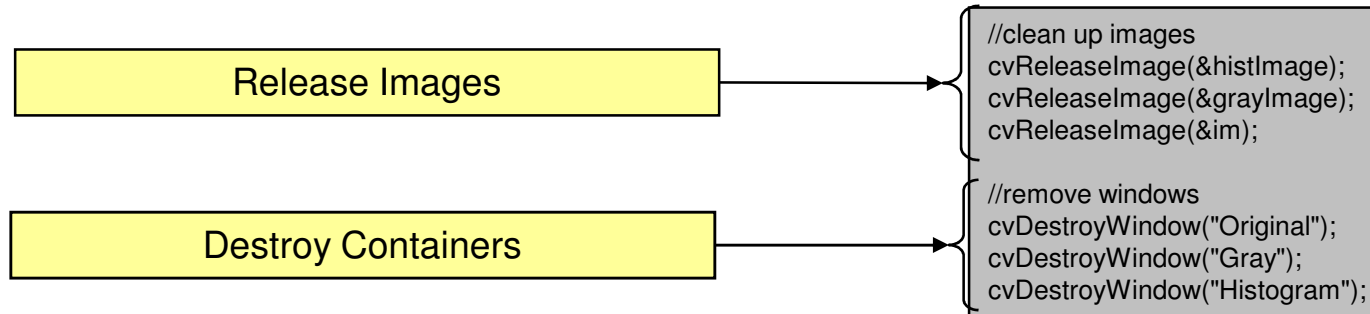
//display the 3 images
cvNamedWindow("Original", 0);
cvShowImage("Original", im );

cvNamedWindow("Gray", 0);
cvShowImage("Gray", grayImage );

cvNamedWindow("Histogram", 0);
cvShowImage("Histogram", histImage );

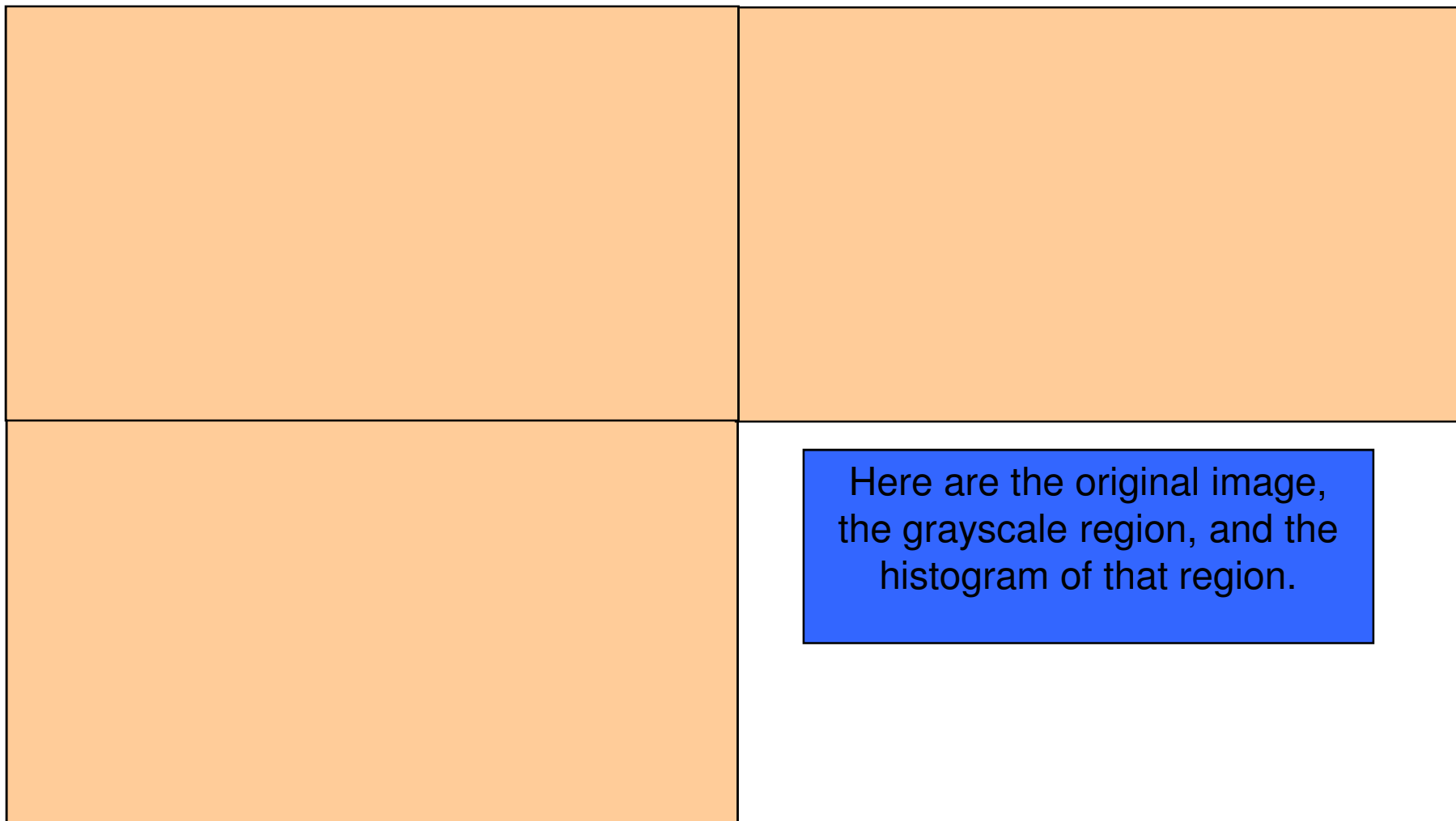
//hold the images until a key is pressed
cvWaitKey(0);
```

# Cleaning Up



It is very important to perform clean-up functions. It is easy for memory utilization to go out of control when multiple images are involved.

# Results



Here are the original image,  
the grayscale region, and the  
histogram of that region.

# Other Histogram Functions

OpenCV has several other functions for working with histograms. These include:

- *cvNormalizeHist*
- *cvThreshHist*
- *cvCompareHist*

For more information about usage of these functions see the OpenCV documentation